Autonomous Agents and Multiagent Systems

2007/2008

Lab 3

Deliberative Agents in the Loading Docks world in NetLogo

1 Objectives

- Implement in NetLogo a practical reasoning agent to solve the *loading docks* world.

2 Exercise

Starting from the base file provided with the lab materials, the student is to implement a BDI (*beliefs*, *desires* and *intentions*) solution for the Loading Docks world.

Chapter 4 of [Wooldridge02] provides the theoretical background for this lab.

The base file contains the following:

- **Control loop –** The basic control loop structure is already implemented. However, some of its key functions are not. This is the objective of this lab. Thus, the agent initially executes a reactive control loop conforming to the previous labs;

- **Beliefs** – The agents' beliefs include information about the map, ramp and shelves. Furthermore, agents' beliefs are already automatically updated;

- **Desires** – The agent should have, at each instant, one of the following desires:
  - *pickup* – the robot wishes to pickup a box;
  - *drop* – the robot wishes to drop the box it's carrying;
  - *return-to-initial-position* – the robot believes all boxes are stored and wishes to return to the initial position;

- **Intentions** – Intentions are based on the desires and determine the agent's behavior. Essentially, desires are converted into intentions which, in turn, lead to the creation of plans. Though primitives to create intentions are provided, converting desires into intentions is not implemented. An intention is a triple with the following elements: <*desire, position, orientation*>. The 'desire' refers to the present desire. The 'position' and 'orientation' refer to the position and orientation which allow the agent to satisfy its desire. For instance, a position and orientation such that it faces a box in a ramp, satisfy the desire to 'pickup'.
- **Plans –** The agents have the capacity to execute plans. A plan is simply a sequence of instructions (as described below). Plan generation is not implemented, however, once create the code executes it automatically;

- **Plan instructions –** Four types of instructions are supported:
  - *instruction-seek-adjacent-position* – Instructs the agent to move to one of the current position's *adjacent* positions (excluding diagonals);
  - *instruction-seek-orientation* – Instructs the agent to turn until facing a specific orientation;
  - *instruction-pickup* – Instructs the agent to pick up the box in front, if it exists;
  - *instruction-drop* – Instructs the agent to drop the box, if an appropriate empty shelf position is in front;

- **Agent communication** – The following communication primitives are provided:
  - *send-message* – sends a message to a specific robot;
  - *broadcast-message* – broadcasts a message to all robots, including the self;
  - *new-message* – handles incoming messages

- **Miscellaneous functions** – Besides data structures functions (intentions, instructions, plans, etc.), several useful auxiliary functions are provided (e.g.: *create-plan-path, position-adjacent-to-occupied-ramp, position-adjacent-to-free-shelf*, etc.).

After understanding the base file, resolve the following exercises:

1. In the previous labs, you've learnt that the reactive agent, though capable of storing the boxes, is unable to return to the initial position. Now that the agents are allowed to have beliefs, implement an *efficient* solution to make them return to the initial position once the boxes have been stored. Assume the agents know, from the start, how many boxes need to be stored (global variable *BOX_COUNT*).
   Note: You can't call the method *boxes-in-shelves*. (Why?)

2. Implement the BDI agent.
   To accomplish this, the student must implement the following functions: *options*, *filter-options* and *create-plan-based-on-intention*. The first determines the current desire, the second converts it into an intention and the third creates a plan to achieve the intention.

3. Make the intentions persist. The agent should not abandon its intentions as soon as an action fails and should seek alternatives. For instance, when two robots collide, both should seek to overcome the collision and proceed trying to achieve the intention. Use the function *collision*, which is immediately called when a collision occurs, to handle collisions.

4. Discuss problems with the deliberative approach to solve the Loading Docks world.

Reference: [Wooldridge02] - Wooldridge, M.; *An Introduction to Multiagent Systems*; John Wiley & Sons, Ltd; 2002